NATIONAL

OSG

NEWSLETTER

Editor : Gordon Bentzen
8 Odin Street,
SUNNYBANK Qld. 4109.
(07) 345-5141

AUGUST 1988

NATIONAL OS9 USER GROUP NEWSLETTER

EDITOR : Gordon Bentzen
HELPERS : Bob Devries and Don Berrie

SUPPORT : Brisbane OS9 Level 2 Users Group.

I am pleased to report that OS9 is indeed alive. Well at least some of you out there seem to support our view as a number of the application forms have been returned complete with the required subscription for one and even two years. Our thanks go to those who have responded so promptly as this is just the encouragement that we need to move on to bigger and better things. If you are anything like me in returning such forms, it is a case of "Round Tuit" one day soon.

The editorial of the July '88 edition outlined some of our plans for this newsletter and perhaps this is a good time to elaborate a little for your information and to answer some questions that have been raised.

Firstly, I must comment on the production of this newsletter. I feel confident that the National OS9 User Group, by way of this newsletter, will now continue to exist due in no small way to the efforts and enthusiasm of Bob Deveries and Don Berrie. Without the help of these two guys, I would not contemplate the task of editor of this monthly production. This is probably best described as a team effort, and that makes it all the more interesting.

The National OS9 User Group newsletter continues to have similar aims as those implemented originally by Graeme Nichols and I am sure that you will agree that Graeme did a great job for a number of years.
This newsletter will be produced on a non-commercial basis purely to share knowledge of the OS9 operating system Basic09 and other languages such as C. We hope to be able to provide tips and general assistance on any software running under OS9 plus help with any hardware which enhances the OS9 operating system and applications.

The July edition was produced from our own resources as stated in that newsletter, and we have elected to also do the same with this edition. Subscriptions already received will be recorded as commencement from 1st.September '88 and in fact, further subscriptions received by the end of August will also be so recorded. The September edition will be mailed only to financial members.

What does the $18.00 subscription cover? This covers the costs of printing and postage for twelve editions, and this will be pretty tight. Only by maintaining the special rates for photocopying we have been able to arrange, will we be able to produce the 10 to 12 page monthly issue, which is our aim. Your contribution of any suitable article, no matter how small, will be a great help. By the way if you have any trouble with programs, code or patches published in this newsletter, you may send us your disk with return postage and a description of your problem. We will analyse the problem and return your disk with the fix and explanation.

Do you have something to advertise? This newsletter will include Ads. for hardware etc. produced by members of this user group. If you produce some item for the CoCo as a hobby or part time activity, then drop me a line. No charge for such ads. will be made and ads from companies or commercial enterprises will NOT be accepted. There is no intent to make the National OS9 User Group a profit making concern in any way.

Will the U.S. User Group disks be available? We do have U.S. OS9 User Group disks of public domain software from Disk 0 to Disk 42, except 27,28, 41. All of this software was acquired prior to the release of the CoCo3 and Tandy OS9 Level 2 but is available to any financial member of the National group. To acquire a copy of any of these disks please mail OS9 formatted disks to the address shown on the newsletter cover together with postage stamps to cover the return postage. A nominal charge of $2-00 per disk is also required to cover the cost of copying and to assist in the purchase of future U.S. User Group disks. We have mailed a request for membership to the U.S. OS9 group and hope to be able to purchase back copies of their disks to get us up to date. We will keep you informed on progress in these pages.

If you would like copies of any of the disks that we do have, please send formatted disks and please advise what format you are using. e.g.CoCo format 40 track Double Sided. The library of U.S. disks is currently stored on 80 track double sided disks. On average each of these disks contains four (4) of the disks as originally supplied by the U.S. group. The $2-00 charge per disk copy will be based on Physical disks. Therefore a backup of one 80 track library disk will cost $2-00. A copy of one (1) U.S. disk on a 35 track single sided disk will also cost $2. The logic used here is that it is probably easier to do a backup of the 80 than to do a file copy to any other format. We are, however, able to handle any format of 5 1/4 floppy including Standard OS9, Mizar, Japan OS9 etc.

As well as seeking contributions of articles etc. for inclusion in these pages we are very interested to know what you would like to see here and what OS9 problems you have that we may be able to help with. Please drop us a line with your comments or problems. We will attempt to print your questions and the answers in the same edition where possible, and so assist as many members as possible. I expect that, at some stage, we will be presented with a problem that we can't answer, and that is where we will have the resources of the entire membership. The special interests noted so far include, OS9 Level 1, OS9 Level 2, 'C', Sculptor, OSK, word processing, and the most common, a general interest in all OS9 topics. What are your special interests?

As stated in our first edition last month, we do need at least 20 members to make this newsletter seem worthwile producing, and we are NOT to that number yet. So if you intend to support this newsletter and just haven't gotten around to it, please do so now. A copy of the application form is again included just in case you may need one for a friend. Thanks again to those who have replied so promptly, we will make this a worthwile project.

Gordon.

****

DYNACALC PATCHES

The Tandy OS9 Dynacalc works great under Level 2 on an 80 column text screen but when I dump a spreadsheet to the printer I have noted a couple of annoying bugs which spoil the hardcopy of the table. I hope the patches below will also help you if you have the same problems.

The following patches will fix 'BUGS' in Version 01.00.00 Dynacalc

One problem with Dynacalc is that it sends an extra LINEFEED with every Carriage Return. This presents a problem with Tandy Printers, and possibly others. To get rid of that double spaced print-out.

Offset	Old	New
0007	80	81
0008	63	62
4BE2	26	20

Problem Two :- Tandy Printers (and probably others) do not recognise the "Top of Form Feed" character that DYNACALC sends out. This results in a form-feed at the very start of a printout. Here is the 'Fix'.

Offset	Old	New
5000	0C	0D

That's all there is to it. I now get a correctly spaced Printout and I don't get a Form Feed at the beginning of each printout. These patches will also correct the same problems in Dynacalc running under OS9 Level One.

For those who are new to OS9 let's go through a step by step procedure using Modpatch under level 2. If you run Level One, you can use the DEBUG command to make the changes, but don't forget to Verify U to update the module CRC

OS9 Level 2 Modpatch of Dynacalc, step by step.
Modpatch will modify modules in memory, so first you must load Dynacalc from your CMDS directory.
We will also assume that your working disk is in drive 0. Your screen should look like the sample below when you have finished. Note that Modpatch will generate some of the display, but don't let that put you off, and of course do hit <ENTER> when you type a command line and nothing happens. Also take care to use spaces between values to observe the correct syntax.

```
OS9:load /d0/cmds/modpatch
OS9:modpatch
l l dynacalc
c c 0007 80 81
0007 80 81
c c 0008 63 62
0008 63 62
c c 4BE2 26 20
4BE2 26 20
c c 5000 0C 0D
5000 0C 0D
v
<ctrl><break>
```

The 'l' links the module 'dynacalc' to modpatch, The 'c' (you only type one of them as the 'l' described above) is the command for Change. The byte address is the one you want to change and often referred to as the offset address, this is followed by 'present value' and 'new value' You must know what the present value is and specify it correctly or Modpatch will give an error. The 'V' as the last command will verify - update the CRC.

When you have completed all the change that you wish to make, the only way out of Modpatch when using from the keyboard, is to send an escape character. i.e.<ctrl><break>

Now that you have patched the module in memory you will want to save it to your CMDS directory. First delete the origional version that you used for the Load.

```
OS9: del /d0/cmds/dynacalc
OS9: save /d0/cmds/dynacalc dynacalc
```

Make sure that you have the SAVE module in your CMDS directory, it does not come with OS9 Level 2 as purchased from Tandy. SAVE is with the Development Pak or the SAVE from Level 1 commands works fine. And of course make sure that you are doing this on a backup disk in case something Goes Wrong, Goes Wrong, Goes Wrong. Wrong.

Here is a dump of the first part of Dynacalc showing a couple of the new values in boldtype.

```
Dump of : /d1/dynacalc
```

Address	0 1	2 3	4 5	6 7	8 9	A B	C D	E F	0 2 4 6 8 A C E
00000000	87CD	5218	000D	1181	6208	2335	0044	796E	.MR.....b.#5.Dyn
00000010	6163	616C	E316	1C76	161C	7316	04BE	1606	acalc..v..s..>..
00000020	3616	05F5	1600	5816	05C2	1602	D716	0951	6..u..X..B..W..Q
00000030	1600	0916	033C	1609	5616	0955	8614	9761	.....<..V..U...a
00000040	170A	8F17	0A6E	0A6A	DC6A	39C6	0AE1	8426	.....n.jF.a.
00000050	1DA6	0184	F7A7	0139	C680	1703	7F96	C910	. ..w'.9F.....I.
00000060	2705	5581	8210	2403	6A8B	9916	04FC	9FAF	'.U...$.j....!./
00000070	1702	C6C0	89D7	B431	C904	89A6	C904	0117	..F@.W41I.. I...
00000080	504B	C680	A7A5	DE1A	33C9	02F8	10DF	CE17	PKF.'%^.3I.x._N.
00000090	2C59	3348	172C	54DE	1A17	05C6	0F76	0FD9	,Y3H.,T^...F.v.Y

```
PROCEDURE iconedit
(* Iconeditor Version 2.03 *)
(* Copyright (c) Feb 1988 D.A.Berrie *)
(* Released to the public domain for personal use only - July 1988 *)

BASE 0
TYPE  rodent=vid,act,totm:BYTE;  x1:INTEGER; ttto:BYTE; tsst:INTEGER; cbsa,cbsb,ccta,cctb,ttsa,ttsb,tlsa,tlsb:BYTE;
x2,bdx,bdy:INTEGER; stat,res:BYTE; AcX,AcY,WrX,WrY:INTEGER
DIM mouse:rodent
(* The above two lines sets up a data area to receive the mouse data *)
DIM pathname,filename:STRING[30]
DIM icon(576),icon$(256):BYTE
DIM i,x,y:INTEGER
DIM c,scrsiz,j,path,path1,callcode:BYTE
DIM flag:BYTE
DIM keypress:STRING[1]
(* Initialize Array Data *)
DATA $20,$43,$6F,$6D,$70,$69,$6C,$65,$64,$20,$75,$73,$69,$6E,$67,$20
DATA $69,$63,$6F,$6E,$20,$65,$64,$69,$74,$6F,$72,$20,$76,$65,$72,$73
DATA $69,$6F,$6E,$20,$32,$2E,$30,$33,$20,$77,$72,$69,$74,$74,$65,$6E
DATA $20,$62,$79,$20,$44,$6F,$6E,$20,$42,$65,$72,$72,$69,$65,$20,$20
scrsiz=2
FOR i=160 TO 223
READ icon$(i)
NEXT i
(* Finished initializing standard data *)

OPEN #path1,"/w":UPDATE Set up graphics screen - next unused descriptor *)
RUN gfx2(path1,"dwset",6,0,0,40,24,1,0,0) Screen type 6=320X192 4-color *)
RUN gfx2(path1,"select") Switch to graphics screen *)
pathname="/d0/cmds/icons/" You may need to change this pathname to suit *)
keypress=""
flag=0
RUN setupmouse(path1) Sets up mouse driver using Hi-Res Interface *)
RUN gfx2(path1,"gcset",202,1) Arrow mouse-pointer *)
RUN gfx2(path1,"color",1,0) Blue arrow on white background *)
RUN clear(icon,icon$)
c=0
1 PRINT #path1,CHR$($0C) Clear Screen *)
PRINT #path1,"<E>dit or <C>reate :"; Set up file to save/edit icon *)
INPUT #path1,keypress
IF keypress="C" OR keypress="c" OR keypress="E" OR keypress="e" THEN
GOTO 2
ELSE
GOTO 1
ENDIF
2 PRINT #path1,
PRINT #path1,
PRINT #path1,"Enter filename :"
INPUT #path1,filename
IF keypress="C" OR keypress="c" THEN
CREATE #path,pathname+filename:UPDATE Create a new file *)
ELSE
OPEN #path,pathname+filename:UPDATE If file already exists open it *)
SEEK #path,0 Move file pointer to start of data *)
GET #path,icon$ Read data if file already exists *)
ENDIF
```

```
FOR i=0 TO 143              - This routine translates the icon data *)
icon(i*4)=INT(icon$(i)/64)    from a single 256 character string to *)
j=MOD(icon$(i),64)            an array of 576 bytes                 *)
icon(i*4+1)=INT(j/16)
j=MOD(j,16)
icon(i*4+2)=INT(j/4)
icon(i*4+3)=MOD(j,4)
NEXT i                        End of translate routine *)
RUN decode(icon,path1) Decodes the array for screen drawing *)
RUN gfx2(path1,"color",1)
RUN gfx2(path1,"curhome")
PRINT #path1,
PRINT #path1,"FILE :        " 9 Blanks *)
PRINT #path1,filename
PRINT #path1,"               " 16 Blanks *)
RUN menusetup(path1)
5 REPEAT   ********    Start of loop which continually reads mouse data *)
RUN getmouse(mouse,path1)
RUN gfx2(path1,"putgc",mouse.AcX/scrsiz,mouse.AcY)
UNTIL mouse.cbsa<>0 End loop when mouse button A is pressed *)
x=3+INT(mouse.AcX/12)*12 Decode mouse data for correct screen size *)
y=2+INT(mouse.AcY/6)*6
RUN limit(x,y,c,flag)
IF flag=1 THEN
icon((x-351)/12+(y-2)*4)=c
RUN gfx2(path1,"color",c) Put data onto screen grid *)
RUN gfx2(path1,"bar",x+1,y-1,x+9,y+3)
GOTO 5
ENDIF
IF flag=2 THEN
FOR i=0 TO 575
icon(i)=0
NEXT i
RUN decode(icon,path1)
RUN menusetup(path1)
GOTO 5
ENDIF
IF flag=3 THEN
j=0
FOR i=0 TO 575 STEP 4 Encode string data  *)
icon$(j)=icon(i)*64+icon(i+1)*16+icon(i+2)*4+icon(i+3)
j=j+1
NEXT i
RUN gfx2(path1,"curon")
RUN gfx2(path1,"color",1)
RUN gfx2(path1,"curxy",0,20)
RUN gfx2(path1,"gcset",0,0) Turn off graphics cursor *)
PRINT #path1,"Save/Rewrite file (Y/N) :"; File routine *)
INPUT #path1,keypress
PRINT #path1,
IF keypress="y" OR keypress="Y" THEN
FOR i=0 TO 223
SEEK #path,i
PUT #path,icon$(i)
NEXT i
SHELL "attr "+pathname+filename+" pe e" Must have execute attributes set *)
ENDIF                                   to allow Multi-Vue to decode it  *)
```

```
CLOSE #path
RUN gfx2(1,"select") Select original window *)
RUN gfx2(path1,"dwend") Remove graphics window *)
CLOSE #path1            and close it          *)
END
ENDIF
GOTO 5

PROCEDURE clear Clears Arrays of extraneous data *)
BASE 0
PARAM icon(576),icon$(576):BYTE
DIM i:INTEGER
FOR i=0 TO 575
icon(i)=0
NEXT i
FOR i=0 TO 143
icon$(i)=0
NEXT i

PROCEDURE getmouse Mouse data - gets this each time mouse is read !!!!! *)
TYPE rodent=vid,act,totm:BYTE; res1:INTEGER; ttto:BYTE; tsst:INTEGER; cbsa,cbsb,ccta,cctb,ttsa,ttsb,tlsa,tlsb:BYTE;
res2,res3,res4:INTEGER; stat,res:BYTE; AcX,AcY,wrx,wry:INTEGER
TYPE registers=cc,a,b,dp:BYTE; x,y,u:INTEGER
DIM regs:registers
DIM callcode:BYTE
PARAM mouse:rodent
PARAM path1:BYTE
regs.a=path1
regs.b=$89
regs.x=ADDR(mouse) Address of data area *)
regs.y=1
callcode=$8D               SS.Mouse System call - see p 8-125 of Manual *)
RUN syscall(callcode,regs) Some versions of the manual have an error in *)
END                        the definition of this mouse packet         *)


PROCEDURE decode Procedure to draw the translated data onto the *)
BASE 0           graphics screen                                 *)
PARAM icon(576),path1:BYTE
DIM x,y,i:INTEGER                    This procedure will still be  *)
PRINT #path1,CHR$($0C)               executed even if a new file   *)
PRINT #path1,"DECODING FILE"         has been created, or the      *)
PRINT #path1,                        screen grid has been cleared. *)
PRINT #path1,"PLEASE WAIT"           In both of the latter cases   *)
RUN gfx2(path1,"curoff")             the screen will be redrawn    *)
RUN gfx2(path1,"box",351,0,639,144)  in the background color and   *)
FOR i=363 TO 627 STEP 12             therefore it will seem that   *)
RUN gfx2(path1,"line",i,0,i,144)     nothing is happening - please *)
NEXT i                               be patient!                   *)
FOR i=6 TO 138 STEP 6
RUN gfx2(path1,"line",351,i,639,i)
NEXT i
i=0
FOR y=2 TO 140 STEP 6
FOR x=352 TO 629 STEP 12
RUN gfx2(path1,"color",icon(i))
RUN gfx2(path1,"bar",x+1,y-1,x+9,y+3)
```

```
(* PROCEDURE decode (cont.) *)
i=i+1
NEXT x
NEXT y

PROCEDURE setupmouse Set mouse resolution and active port routine *)
PARAM path1:BYTE
TYPE registers=cc,a,b,dp:BYTE; x,y,u:INTEGER
DIM regs:registers
DIM callcode:BYTE
regs.a=path1
regs.b=$94
regs.x=$0101 Hi.res mouse, Right port *)
regs.y=$FFFF No changes to key repeat *)
callcode=$8E
RUN syscall(callcode,regs) SS.GIP System Call - sets the system wide *)
END                        Mouse and Key Repeat parameters          *)

PROCEDURE limit      This procedure checks to see that the mouse button *)
PARAM x,y:INTEGER    can only be read when the data is valid            *)
PARAM c:BYTE
PARAM flag:BYTE
DIM i:INTEGER
flag=0
IF x>635 THEN
x=628
ENDIF
IF x>339 THEN
flag=1 Mouse pointer within the screen grid *)
ENDIF
IF x<339 THEN
FOR i=0 TO 3
IF x>10+i*80 AND x<60+i*80 AND y>60 AND y<75 THEN
c=i
ENDIF
NEXT i
ENDIF
IF x>10 AND x<100 AND y>100 AND y<115 THEN
flag=2 Mouse pointer in palette area *)
ENDIF
IF x>210 AND x<300 AND y>100 AND y<115 THEN
flag=3 Mouse pointer in CLEAR/QUIT area *)
ENDIF
IF y>140 THEN
y=140
ENDIF
END

PROCEDURE menusetup Procedure to set up left side of screen *)
PARAM path1:BYTE
DIM x,y,i,j:INTEGER
PRINT #path1,
PRINT #path1,"Palette Selection"
x=10
y=60
j=0
```

```
FOR i=0 TO 3
RUN gfx2(path1,"color",1)
RUN gfx2(path1,"box",x+i*80,y,x+i*80+50,y+15)
RUN gfx2(path1,"color",j)
RUN gfx2(path1,"bar",x+i*80+4,y+2,x+47+i*80,y+13)
j=j+1
NEXT i
RUN gfx2(path1,"color",1)
x=10
y=100
RUN gfx2(path1,"box",x,y,x+90,y+15)
RUN gfx2(path1,"curxy",1,13)
PRINT #path1,"CLEAR"
x=210
RUN gfx2(path1,"box",x,y,x+90,y+15)
RUN gfx2(path1,"curxy",14,13)
PRINT #path1,"QUIT"
END

 (* End of Basic09 File *)
```

EDITORS NOTE

When typing this program, you will need to add the string "(*" at the beginning of most of the comment lines. Our printer seems unable to handle that particular combination of characters.
Please accept our apologies.

ooooo00000=====00000ooooo

This is a fairly rough and ready editor for Multi-Vue, Tandy's mouse driven environment. As you can see it is written in Basic09, and while it works, it does have a few drawbacks. Firstly, there is a fair amount of number crunching involved in the string translation routines. That fact, added to the requirement to use the GFX2 bar-drawing calls to actually draw the enlarged icon on the screen, results in a product which is a bit slow. I guess if you are familiar with Multi-Vue you will be used to that anyway.

An easy way to improve the speed would be to rewrite the number crunching routines in either "C", or assembly language. The actual screen drawing routines could be speeded up using a memory mapped screen (as in Rogue, Koronis Rift etc.), but I imagine that Basic09 would not be a very efficient way to do this. So I guess, until someone comes along with a better one, this will have to do.

There are a couple of things to be aware of in order to get the program functioning. The program will look in the current execution directory for the GFX2 and SYSCALL routines, and will not run unless it finds them. It will also look for the general window descriptor "/w", so that it can open the next available window. That means that you must have both the "/w" descriptor and at least one free window descriptor available in memory. If not you will have to either incorporate them in your boot, or load them from disk before you start. The final files need to have the execute attributes set in order to allow Multi-Vue to decode them, and therefore the program needs to have access to the ATTR command.

It may be necessary for you to alter the pathname for the directory in which Multi-Vue will look for your icons. I have flagged this in the source so that you can do so if you wish. If you do not have a hi-res mouse/joystick interface, you will need to either use the keyboard mouse (press <CTRL> + <CLEAR>; Use the arrow keys to move the cursor and the function keys for the fire buttons) or change the line :

```
regs.x=$0101 to regs.x=$0001
```

in the procedure SETUPMOUSE.

If for any reason the program crashes, it will close the graphics window and switch to the next window, rather than to the window that it was created from. I am not sure how to make it switch back to its original window on a crash. Perhaps someone else could enlighten me !

Should you have any problems with this program please contact me and I will try to assist you. There are a number of ways in which the program could be enhanced and please feel free to modify the source in any way that you wish.

Don Berrie (07) 375-3236

USER FRIENDLY WINDOWS

The following program is a short program written in "C" to demonstrate the use of the windowing functions available in the CGFX.LIB library which comes with Multi-Vue.

I have included it purely as a demonstartion ... it really needs to be refined in a number of ways. All that the program really does is to open a symmetrical overlay window on an 80X24 screen. It will allow you to select the foreground and background colours, and the top left hand corner.

WARNING: There is no error checking. If you happen to nominate a starting position far enough down, or across the screen, so that it cannot open a window it will cause the system to CRASH!!!

When you try to compile this program, or indeed any C program which requires the use of the SYS.LIB or the CGFX.LIB libraries, you will need to use the rlink linker which is supplied on the Multi-Vue disk. Consequently, you will need to compile it using the "-r" option, and manually link it. Try the following line :
RLINK /d1/lib/cstart.r ctmp.3.r -o=window -l=/d1/lib/cgfx.l -l=/d1/lib/sys.l -l=/d1/lib/clib.l

If you do not have a C compiler, try to locate a fellow user who has one, or if all else fails, send me a disk with return postage, and I will return it with a compiled version for you.

Best of luck Don Berrie.

```
#include <wind.h>
main(argc,argv)
int argc;
char *argv[];
{
    int path,savescn;
    int spx,spy;
    int szx,szy;
    int fprn,bprn;
    if (argc!=4) {
        printf("Usage: window x y z);
        printf("Where x is starting x co-ordinate);
        printf("      y is starting y co-ordinate);
        printf("  and z is background Palette Number);
        exit(0);
    }
    path=savescn=1;
    bprn=atoi(argv[3]);
    if (bprn != 2) {
        fprn=2;
    }
    else {fprn=0;}
    spx=atoi(argv[1]);
    spy=atoi(argv[2]);
    szx=78-spx*2;
    szy=22-spy*2;
    OWSet(path,savescn,spx,spy,szx,szy,fprn,bprn);
```

NATIONAL OS9 USER GROUP NEWSLETTER

NATIONAL OS9 USER GROUP APPLICATION FORM

****

Surname : _____ First Name :_____ Title (Mr.,Dr.,etc) : ____
Street : _____
Suburb : _____ State :___ Postcode : ____

Home Phone :_____ Business Phone : _____

Do you run OS9 Level 1 [_] and/or OS9 Level 2 [_] (please tick)

Type of Computer for OS9 :_____Type of system :_____

Diskette Size (incnes): __ Number of Cylinders (tracks per side): __ Sides:__ Number of Drives :__

Printer Type: _____

Modem Type : _____

Other hardware : _____

Special Interests : _____

_____

Can you contribute articles to this Newsletter :_____

Date : __/__/__ Signature :_____

Amount Enclosed: $ __.__ ($18-00 will cover you for 12 months)

Please Return Completed Form to :-

NATIONAL OS9 USER GROUP
C/o Gordon BENTZEN
8 ODIN STREET,
SUNNYBANK QLD 4109